TECHNICAL UNIVERSITY OF CLUJ-NAPOCA ROMANIA

TECHNICAL UNIVERSITY OF CLUJ-NAPOCA ROMANIA

PROJECT ”E-COMMERCE”

CLASS: OBJECT ORIENTED PROGRAMMING

STUDENTS: SAVU COSMIN

AND

ȘICHET DARIUS

GROUP:30421

PROFESSOR: ARON BAKA BALINT

TABLE OF CONTENTS

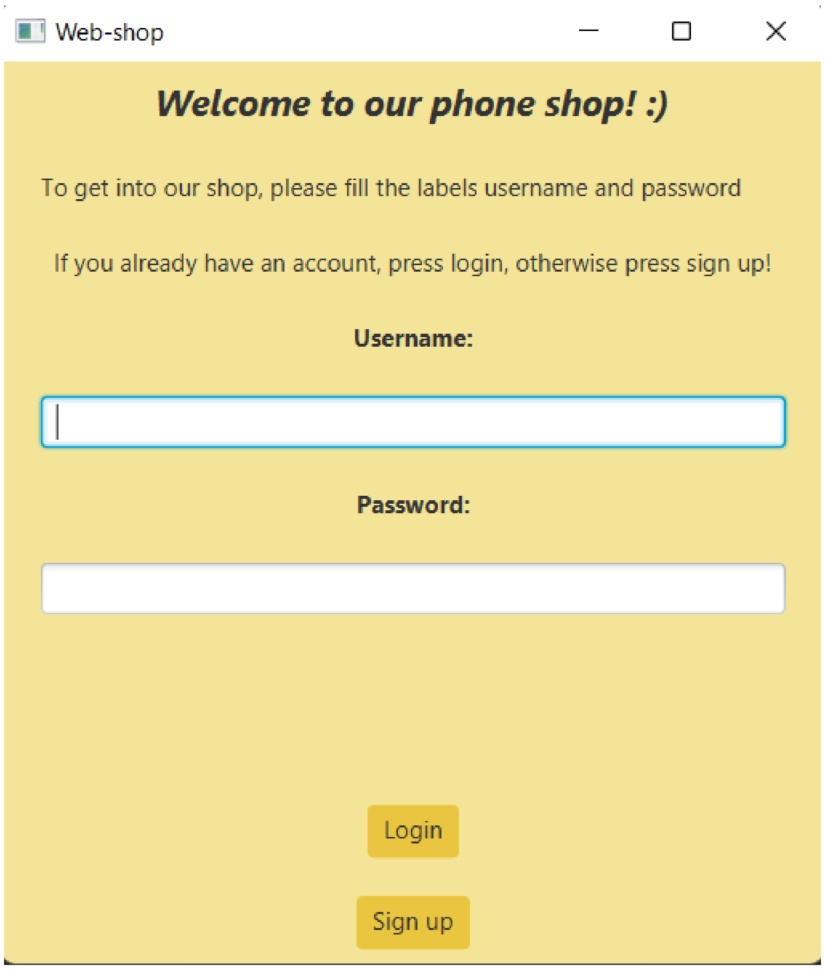
SPECIFICATION

The purpose of this project is to design an E-Commerce store which may be applicable in real world. That being said, the store will be a phone store through which users will be able to purchase their desired phones by registering on the platform. The store's interface has an easy-to-understand structure, so it can be used by any user. This store was created using a database for storing information, but also Java FX and Scene Builder for defining interfaces.

DESIGN, DESCRIPTION AND USE CASES

USER INSTRUCTIONS

The idea behind this phone shop is to provide an easy way for the customers to purchase the phones they desire. In this sense, the homepage was designed, called login view.



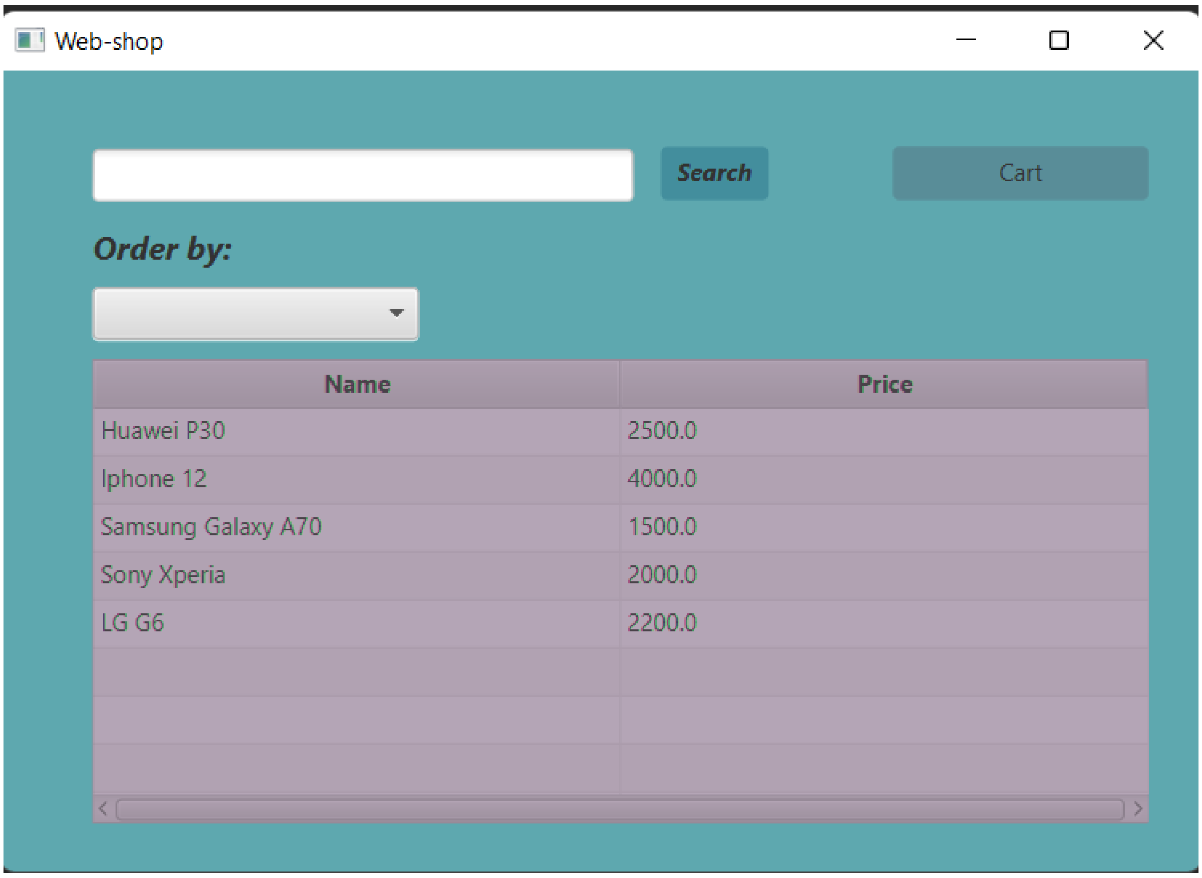
Welcome to our phone shop! :)

To get into our shop, please fill the labels username and password

If you already have an account, press login, otherwise press sign up!

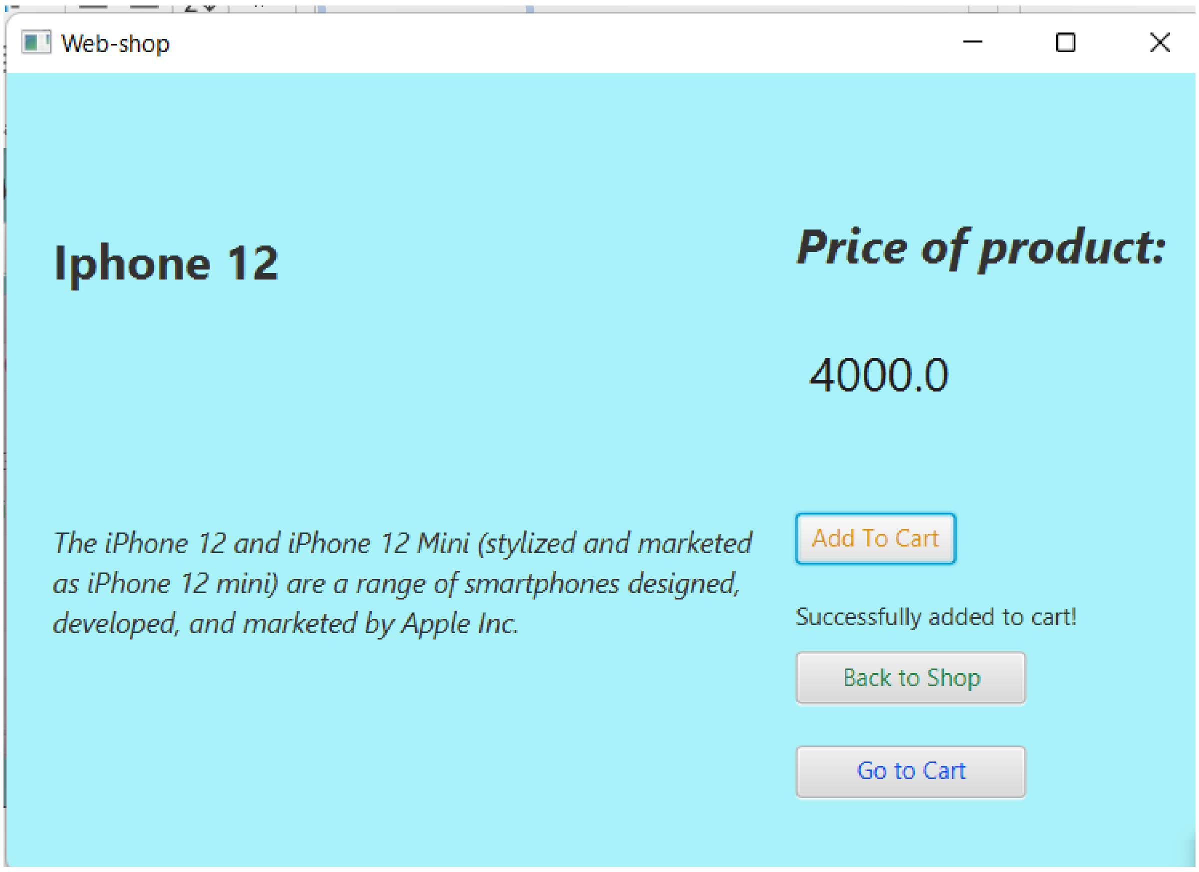
Here, the users can enter the store by filling the labels Username and Password. If the user already has an account, then will press the button login to get into the shop, otherwise will press the button Sign up to create a new account and get into the shop. In the case someone introduces a wrong username or password, a message will be displayed showing that an error has occurred.

Afterwards, the shop view will be displayed, giving access to the products available and listed on the shop.



Name	Price
Huawei P30	2500.0
Iphone 12	4000.0
Samsung Galaxy A70	1500.0
Sony Xperia	2000.0
LG G6	2200.0

Here, the user can search a certain object by searching its name, or order the list of products by choosing the preferred option: Order by price ascending, Order by price descending or display the products in alphabetical order. Then, one can enter in the product view by clicking on the name of the product of interest.


Web-shop

Iphone 12

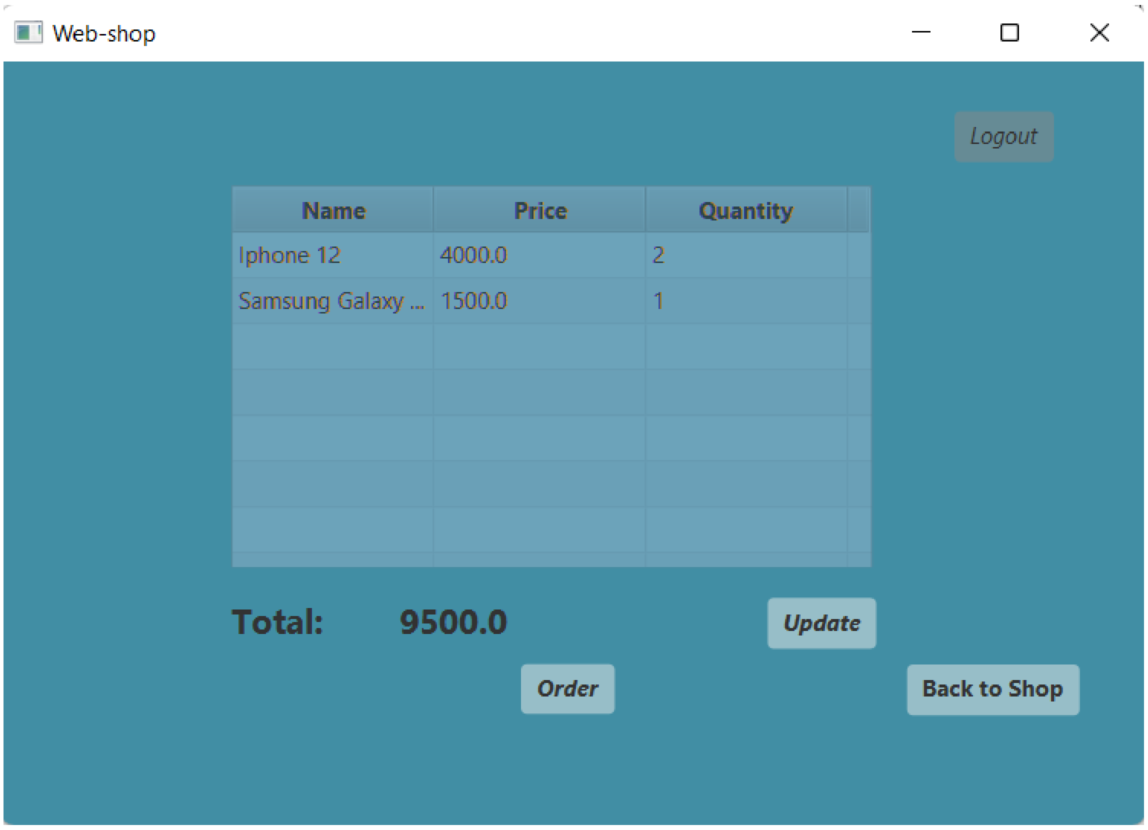
Price of product:

4000.0

The iPhone 12 and iPhone 12 Mini (stylized and marketed as iPhone 12 mini) are a range of smartphones designed, developed, and marketed by Apple Inc.

Successfully added to cart!

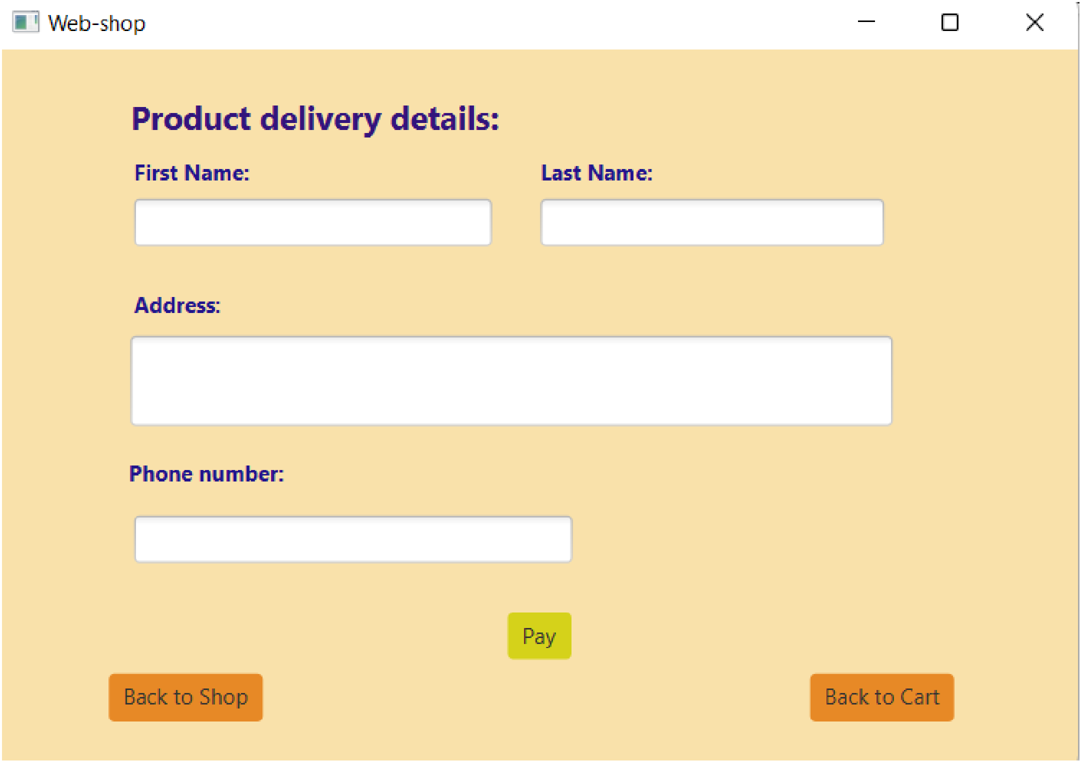
Here, the user can add a certain product to its cart and by pressing the button “Add to cart”, or can go back to the shop view and cart view by pressing the buttons “Back to Shop”, if the customer is interested in other products. If the user wants to enter the cart, he can easily access it by pressing the “Go to cart” button.



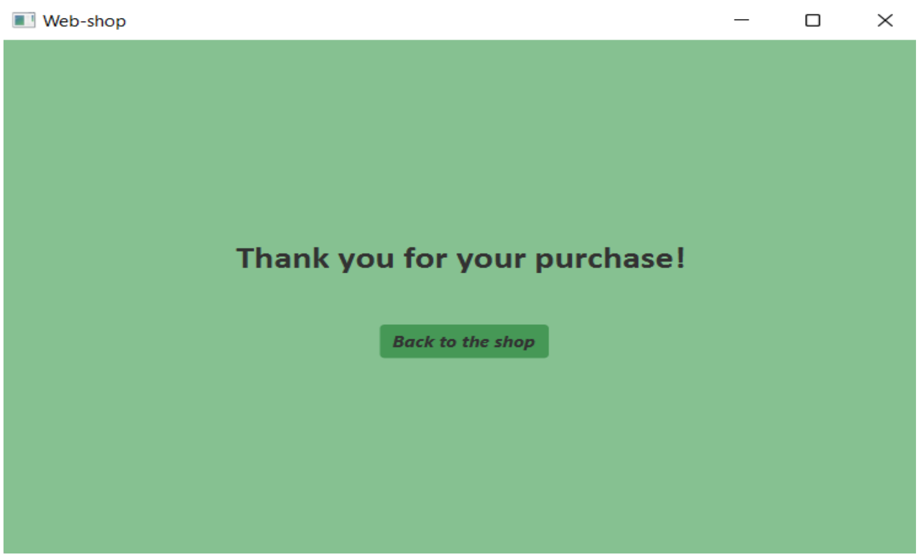
Name	Price	Quantity
Iphone 12	4000.0	2
Samsung Galaxy ...	1500.0	1

Total: 9500.0

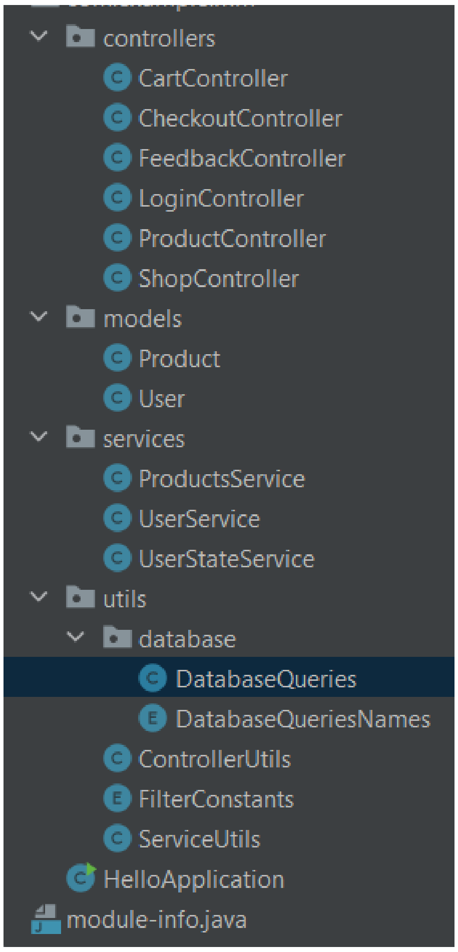
In the cart view, the user can change the quantity of the products they want. That being said, if the user changes the quantity and presses the update button, the total price of the products will change, and if he changes the quantity of a product to 0, that product will disappear from the cart. After pressing the “Order” button, a new window will open, where the customer can fill the labels with information regarding the shipping.



From this window, the customer can go both in the cart or shop, or he can buy the objects selected by pressing buy. By pressing the pay button, a new window is activated, representing the feedback view.



STRUCTURE AND CODE PRESENTATION


```
controllers
    CartController
    CheckoutController
    FeedbackController
    LoginController
    ProductController
    ShopController
models
    Product
    User
services
    ProductsService
    UserService
    UserStateService
utils
    database
        DatabaseQueries
        DatabaseQueriesNames
    ControllerUtils
    FilterConstants
    ServiceUtils
HelloApplication
module-info.java
```

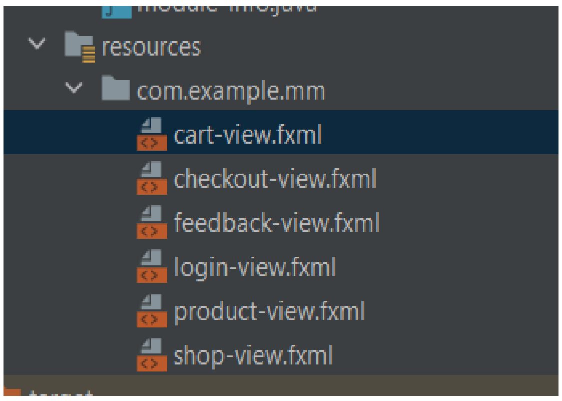
The structure of the shop is based on 4 categories:

-the controllers, which control the actions performed on windows

-the models- which define the models used for this project: the products and the customers

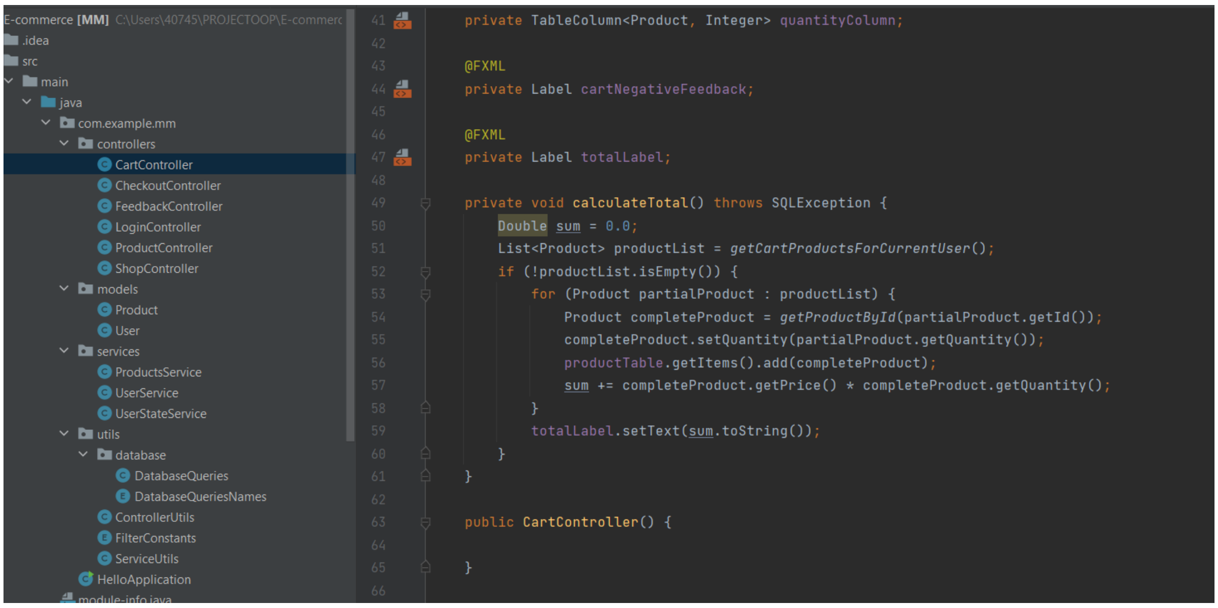
-the services- which realize the connection between the shop and the data base where the information is stored

-the database queries: where the commands used from the database are stored.


```
resources
    com.example.mm
        cart-view.fxml
        checkout-view.fxml
        feedback-view.fxml
        login-view.fxml
        product-view.fxml
        shop-view.fxml
```

The views from Java FX (and with the use of Scene Builder) – which help to create the interfaces of the pages, as well as having an overview of the appearance

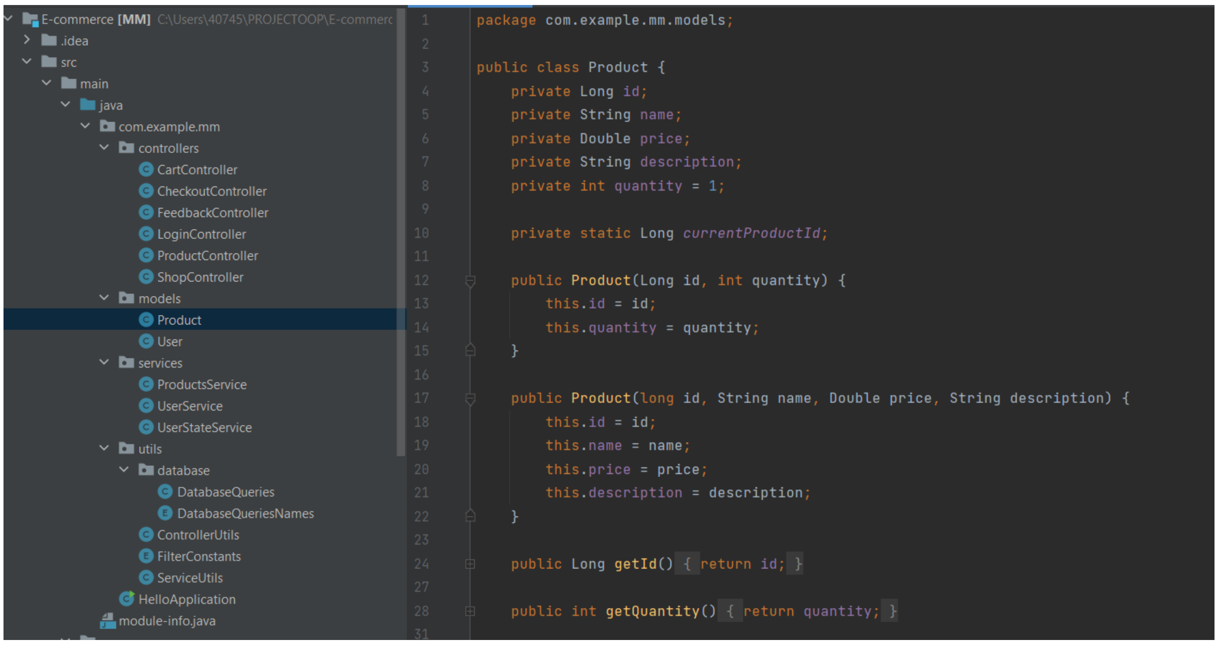
Controller's example:



```
E-commerce [MM]  C:\Users\40745\PROJECTOOP\E-commerc
  .idea
  src
v   main
  v   java
    v   com.example.mm
      v   controllers
            CartController
            CheckoutController
            FeedbackController
            LoginController
            ProductController
            ShopController
      v   models
            Product
            User
      v   services
            ProductsService
            UserService
            UserStateService
      v   utils
        v   database
              DatabaseQueries
              DatabaseQueriesNames
            ControllerUtils
            FilterConstants
            ServiceUtils
          HelloApplication
        module-info.java
```

```java
    private TableColumn<Product, Integer> quantityColumn;

    @FXML
    private Label cartNegativeFeedback;

    @FXML
    private Label totalLabel;

    private void calculateTotal() throws SQLException {
        Double sum = 0.0;
        List<Product> productList = getCartProductsForCurrentUser();
        if (!productList.isEmpty()) {
            for (Product partialProduct : productList) {
                Product completeProduct = getProductById(partialProduct.getId());
                completeProduct.setQuantity(partialProduct.getQuantity());
                productTable.getItems().add(completeProduct);
                sum += completeProduct.getPrice() * completeProduct.getQuantity();
            }
            totalLabel.setText(sum.toString());
        }
    }

    public CartController() {

    }
```

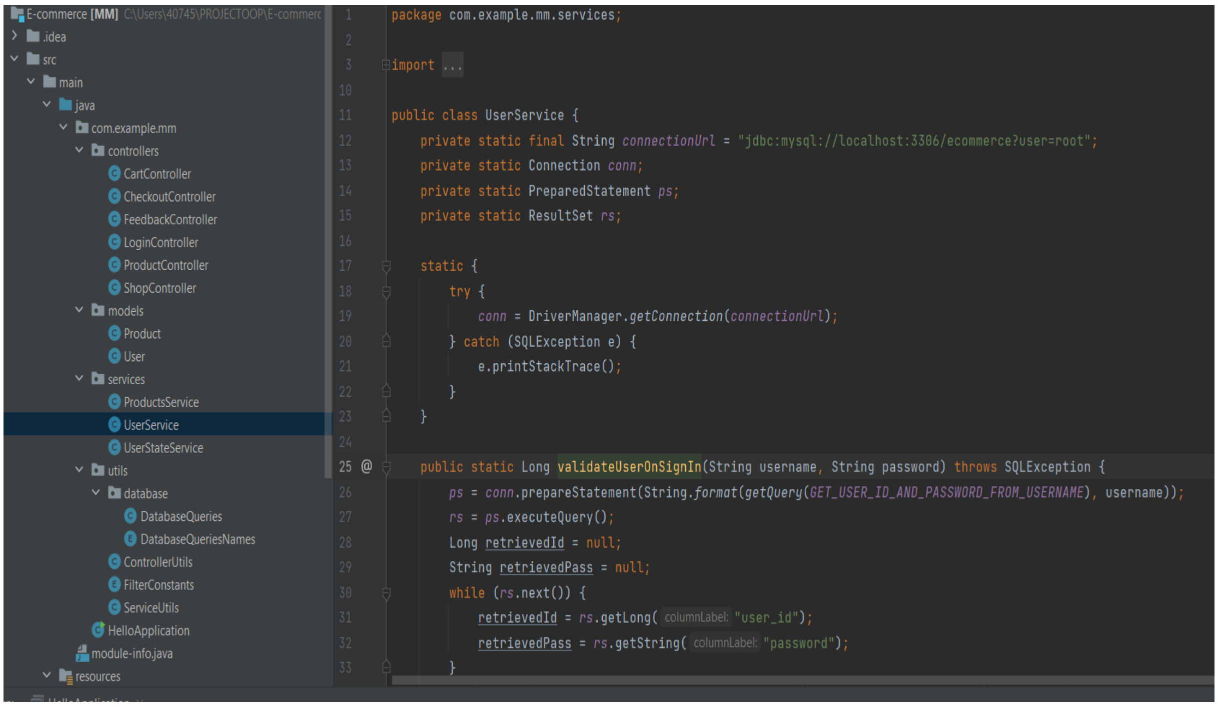
Model example:



```
v E-commerce [MM]  C:\Users\40745\PROJECTOOP\E-commerc
  > .idea
  v src
    v main
      v java
        v com.example.mm
          v controllers
                CartController
                CheckoutController
                FeedbackController
                LoginController
                ProductController
                ShopController
          v models
                Product
                User
          v services
                ProductsService
                UserService
                UserStateService
          v utils
            v database
                  DatabaseQueries
                  DatabaseQueriesNames
                ControllerUtils
                FilterConstants
                ServiceUtils
              HelloApplication
            module-info.java
```

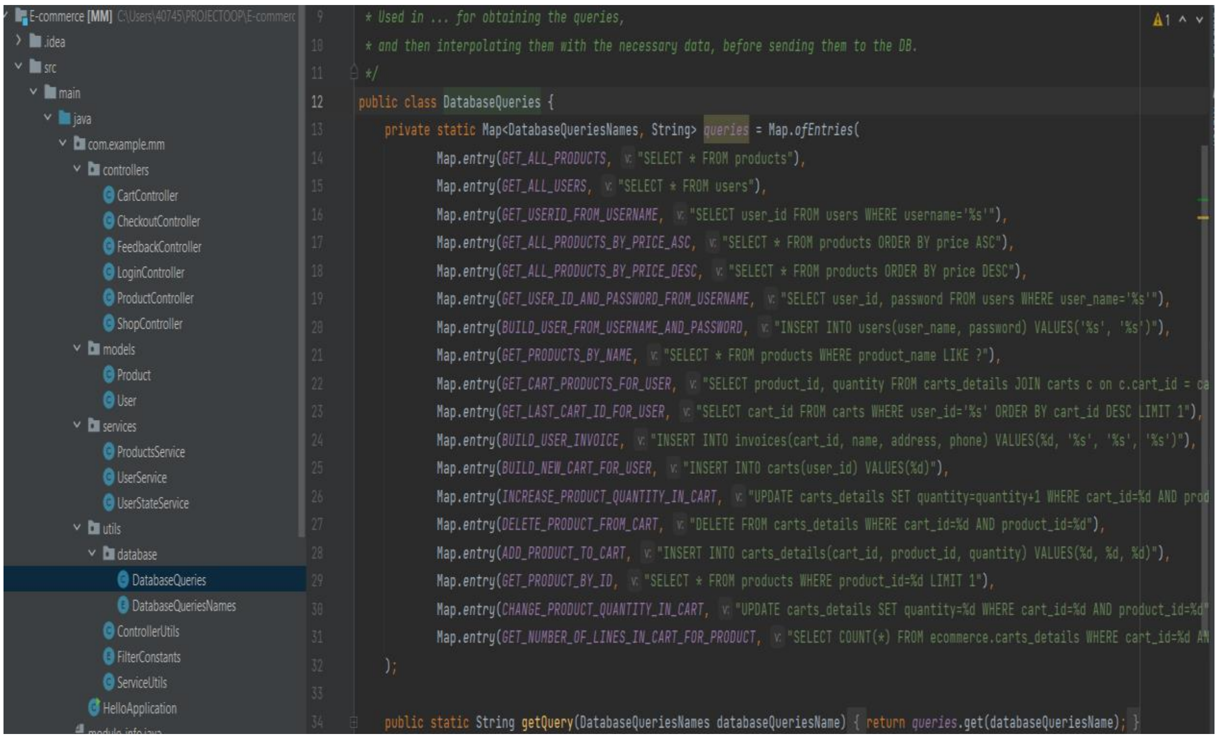
```java
package com.example.mm.models;

public class Product {
    private Long id;
    private String name;
    private Double price;
    private String description;
    private int quantity = 1;

    private static Long currentProductId;

    public Product(Long id, int quantity) {
        this.id = id;
        this.quantity = quantity;
    }

    public Product(long id, String name, Double price, String description) {
        this.id = id;
        this.name = name;
        this.price = price;
        this.description = description;
    }

    public Long getId() { return id; }

    public int getQuantity() { return quantity; }
```

Service Example:



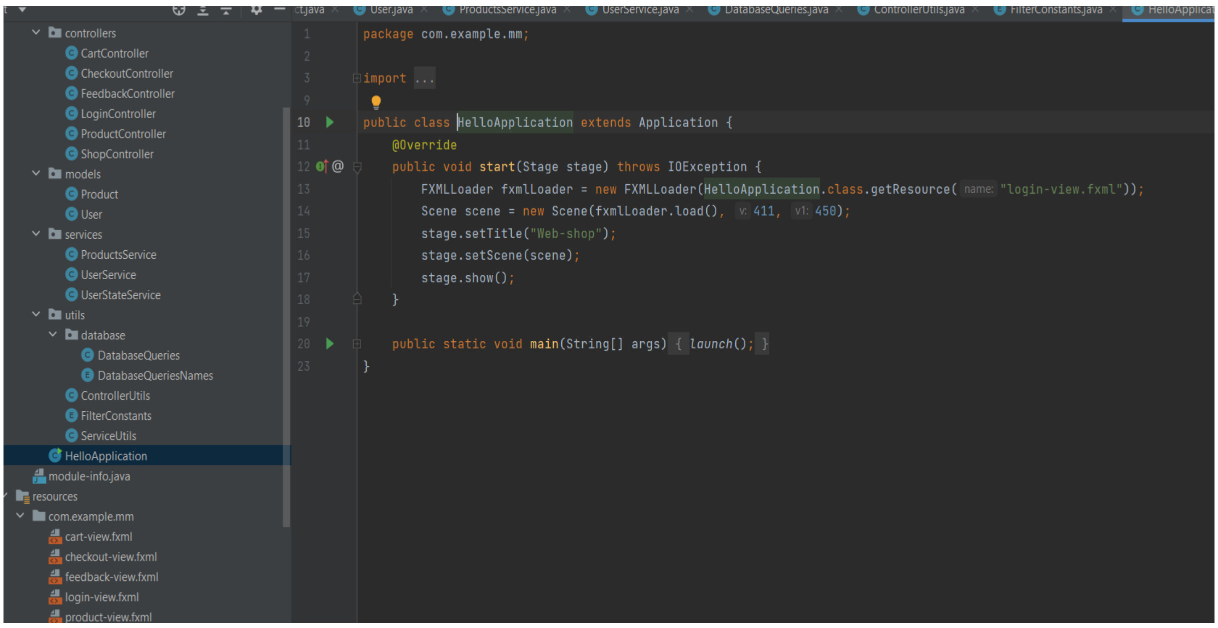
```java
package com.example.mm.services;

import ...

public class UserService {
    private static final String connectionUrl = "jdbc:mysql://localhost:3306/ecommerce?user=root";
    private static Connection conn;
    private static PreparedStatement ps;
    private static ResultSet rs;

    static {
        try {
            conn = DriverManager.getConnection(connectionUrl);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static Long validateUserOnSignIn(String username, String password) throws SQLException {
        ps = conn.prepareStatement(String.format(getQuery(GET_USER_ID_AND_PASSWORD_FROM_USERNAME), username));
        rs = ps.executeQuery();
        Long retrievedId = null;
        String retrievedPass = null;
        while (rs.next()) {
            retrievedId = rs.getLong( columnLabel: "user_id");
            retrievedPass = rs.getString( columnLabel: "password");
        }
```

Database query example:



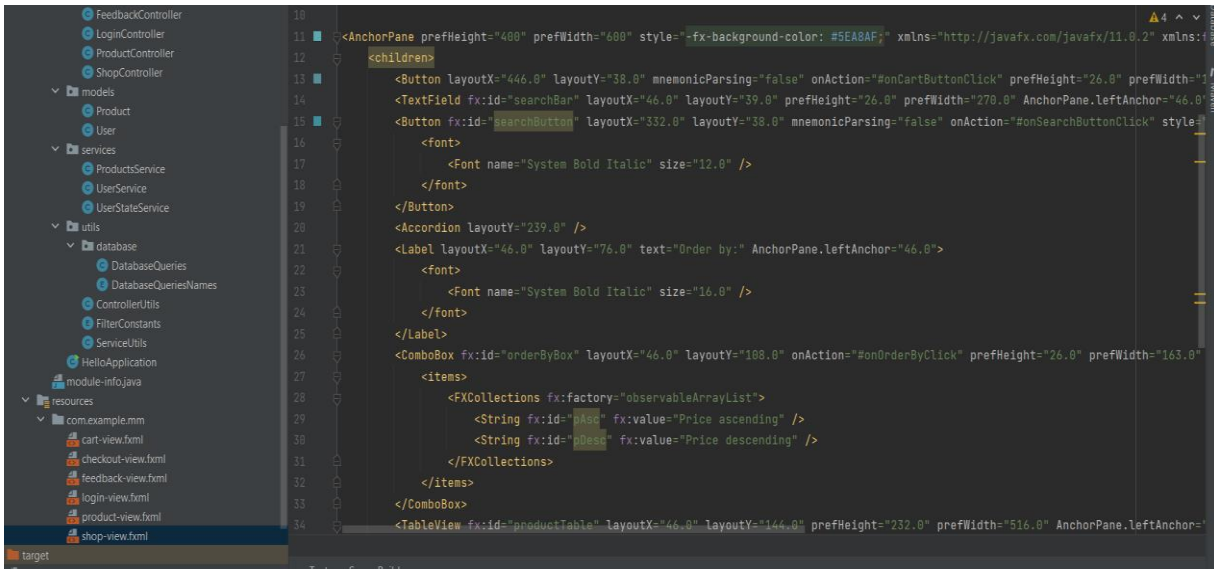
```java
 * Used in ... for obtaining the queries,
 * and then interpolating them with the necessary data, before sending them to the DB.
 */
public class DatabaseQueries {
    private static Map<DatabaseQueriesNames, String> queries = Map.ofEntries(
            Map.entry(GET_ALL_PRODUCTS,  v: "SELECT * FROM products"),
            Map.entry(GET_ALL_USERS,  v: "SELECT * FROM users"),
            Map.entry(GET_USERID_FROM_USERNAME,  v: "SELECT user_id FROM users WHERE username='%s'"),
            Map.entry(GET_ALL_PRODUCTS_BY_PRICE_ASC,  v: "SELECT * FROM products ORDER BY price ASC"),
            Map.entry(GET_ALL_PRODUCTS_BY_PRICE_DESC,  v: "SELECT * FROM products ORDER BY price DESC"),
            Map.entry(GET_USER_ID_AND_PASSWORD_FROM_USERNAME,  v: "SELECT user_id, password FROM users WHERE user_name='%s'"),
            Map.entry(BUILD_USER_FROM_USERNAME_AND_PASSWORD,  v: "INSERT INTO users(user_name, password) VALUES('%s', '%s')"),
            Map.entry(GET_PRODUCTS_BY_NAME,  v: "SELECT * FROM products WHERE product_name LIKE ?"),
            Map.entry(GET_CART_PRODUCTS_FOR_USER,  v: "SELECT product_id, quantity FROM carts_details JOIN carts c on c.cart_id = ca
            Map.entry(GET_LAST_CART_ID_FOR_USER,  v: "SELECT cart_id FROM carts WHERE user_id='%s' ORDER BY cart_id DESC LIMIT 1"),
            Map.entry(BUILD_USER_INVOICE,  v: "INSERT INTO invoices(cart_id, name, address, phone) VALUES(%d, '%s', '%s', '%s')"),
            Map.entry(BUILD_NEW_CART_FOR_USER,  v: "INSERT INTO carts(user_id) VALUES(%d)"),
            Map.entry(INCREASE_PRODUCT_QUANTITY_IN_CART,  v: "UPDATE carts_details SET quantity=quantity+1 WHERE cart_id=%d AND prod
            Map.entry(DELETE_PRODUCT_FROM_CART,  v: "DELETE FROM carts_details WHERE cart_id=%d AND product_id=%d"),
            Map.entry(ADD_PRODUCT_TO_CART,  v: "INSERT INTO carts_details(cart_id, product_id, quantity) VALUES(%d, %d, %d)"),
            Map.entry(GET_PRODUCT_BY_ID,  v: "SELECT * FROM products WHERE product_id=%d LIMIT 1"),
            Map.entry(CHANGE_PRODUCT_QUANTITY_IN_CART,  v: "UPDATE carts_details SET quantity=%d WHERE cart_id=%d AND product_id=%d"
            Map.entry(GET_NUMBER_OF_LINES_IN_CART_FOR_PRODUCT,  v: "SELECT COUNT(*) FROM ecommerce.carts_details WHERE cart_id=%d AN
    );

    public static String getQuery(DatabaseQueriesNames databaseQueriesName) { return queries.get(databaseQueriesName); }
```

Main:

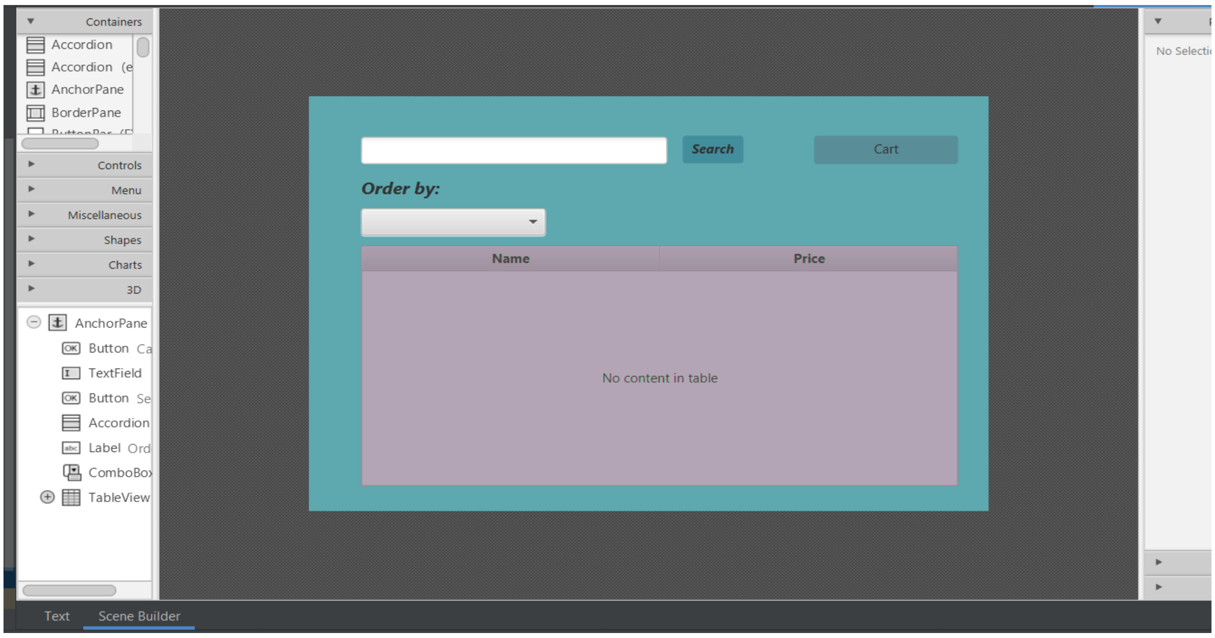


```java
package com.example.mm;

import ...

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource( name: "login-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), v: 411, v1: 450);
        stage.setTitle("Web-shop");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) { launch(); }
}
```

View example:



```xml
<AnchorPane prefHeight="400" prefWidth="600" style="-fx-background-color: #5EA8AF;" xmlns="http://javafx.com/javafx/11.0.2" xmlns:f
    <children>
        <Button layoutX="446.0" layoutY="38.0" mnemonicParsing="false" onAction="#onCartButtonClick" prefHeight="26.0" prefWidth="1
        <TextField fx:id="searchBar" layoutX="46.0" layoutY="39.0" prefHeight="26.0" prefWidth="270.0" AnchorPane.leftAnchor="46.0"
        <Button fx:id="searchButton" layoutX="332.0" layoutY="38.0" mnemonicParsing="false" onAction="#onSearchButtonClick" style="
            <font>
                <Font name="System Bold Italic" size="12.0" />
            </font>
        </Button>
        <Accordion layoutY="239.0" />
        <Label layoutX="46.0" layoutY="76.0" text="Order by:" AnchorPane.leftAnchor="46.0">
            <font>
                <Font name="System Bold Italic" size="16.0" />
            </font>
        </Label>
        <ComboBox fx:id="orderByBox" layoutX="46.0" layoutY="108.0" onAction="#onOrderByClick" prefHeight="26.0" prefWidth="163.0"
            <items>
                <FXCollections fx:factory="observableArrayList">
                    <String fx:id="pAsc" fx:value="Price ascending" />
                    <String fx:id="pDesc" fx:value="Price descending" />
                </FXCollections>
            </items>
        </ComboBox>
        <TableView fx:id="productTable" layoutX="46.0" layoutY="144.0" prefHeight="232.0" prefWidth="516.0" AnchorPane.leftAnchor="
```



FURTHER IMPROVEMENTS FOR THIS PROJECT

One possible improvement for this project could be the addition of images for each product as well as some changes to the layout of the pages so that they look friendlier.